MRS in the LinGO Grammar Matrix: A Practical User's Guide

Dan Flickinger, Emily M. Bender, and Stephan Oepen

December 15, 2003

Contents

1 Introduction **3**

2 Background and Motivation **3**

3 Basic Semantic Objects **4**
3.1 *mrs* . . . 5
3.2 *relation* . . . 5
3.3 *qeq* . . . 8
3.4 *hook* . . . 9
3.5 *semarg*, *tam*, and *png* . . . 10
3.6 *semsort* . . . 12
3.7 *keys* . . . 13
3.8 *lexkeys* . . . 13
3.9 Summary . . . 15

4 Relations and Argument Features **15**
4.1 Ordinary Predicates . . . 16
4.2 Special Relations . . . 19

5 Features of Indices: Agreement and TAM **23**

6 Syntax-Semantics Interface **24**
6.1 Semantic Principles . . . 24
6.2 HOOK Features and Semantic Heads . . . 26
6.3 Semantic Contributions of Constructions . . . 27
6.4 Linking . . . 29
6.5 Indices Bound by Quantifiers . . . 31
6.6 Imposing Handle Constraints . . . 31
6.7 Syntax-Semantics Mismatches . . . 33

7 Verifying *mrs*s **37**

8 Sem-I: The Semantic Interface **38**
8.1 The meta-level Sem-I 38
8.2 The object-level Sem-I 39
8.3 Thematic role mapping 39
8.4 Word classes 39
8.5 Example sentences 40

9 Conclusion **41**

10 Acknowledgments **41**

1 Introduction

This paper is intended to serve as documentation for the semantic aspects of the LinGO Grammar Matrix. We assume familiarity with the specification of Minimal Recursion Semantics (MRS) given in Copestake et al. 2003 (see also Copestake et al. 2001), and focus instead on the practical aspects of writing grammars that produce well-formed *mrs*s and of developing semantic representations of particular linguistic phenomena within MRS.

Section 2 gives some background on the LinGO Grammar Matrix and MRS. Sections 3–6 describe the implementation of MRS in the Matrix and how to extend it in building a grammar of a specific language. Along the way, we provide touchstone linguistic examples and advice on best practice. Section 7 describes tools and methods for verifying the well-formedness of *mrs*s. Finally, Section 8 describes the Sem-I ('Semantic Interface'), a component of the grammar that specifies how *mrs*s are to be interpreted.

2 Background and Motivation

The Matrix grammar starter-kit (Bender et al. 2002) is a language-independent core grammar designed to facilitate the rapid initial development of grammars for natural languages, with foundations solid enough to support steady expansion to broad coverage of the linguistic phenomena in these languages. Such grammars are particularly valuable because they can assign semantic representations to linguistic input, providing the foundation for applications which require natural language understanding. As such, a central component of the Matrix is the collection of resources it contains for simplifying the implementation of semantic composition within each language and supporting the development of a standardized description language for meaning representations, which can provide an effective interface for practical applications. The resources in the Matrix further enable the meaning representations to keep pace as the syntactic analyses of a grammar grow in complexity.

The goal of the Matrix grammar starter-kit is to provide the necessary definitions of core linguistic types for words and phrases at a level of generality which enables quick specialization to encode the additional basic grammatical constraints for a particular language. With this language-specific tuning, it should be possible to construct a grammar within an afternoon which can be used to parse non-trivial sentences of a given language, then use that same implementation as the basis for the development over time of a semantically precise, broad-coverage grammar. The existing Matrix release also includes software links and parameter settings for one particular grammar development system, the LKB (Copestake 2002), which includes an efficient parser and generator, but grammars built on the Matrix can be read and used by a number of other parsers (cf. Oepen et al. 2002).

The Matrix is constructed within the formal system of typed feature structures defined in (Carpenter 1992), using the single operation of unification to build phrases from the words and phrases they contain. Minimal Recursion Semantics was designed to enable semantic composition using only this same unification of typed feature structures, producing for each phrase or sentence a description of the meaning representation sufficient to support logical inference. The type definitions for signs in the Matrix include a semantic component which is an implementation of MRS, and more specifically of the elaboration of a semantic algebra

for MRS presented in Copestake et al. 2001. In addition, MRS was designed to answer the competing demands of expressive adequacy and computational tractability, as well as to allow underspecification where it facilitates computational applications, such as machine translation. Thus Matrix-derived grammars are not only interesting as a means of testing linguistic hypotheses, but also have the potential to be integrated in applications which require natural language understanding, including machine translation, automated email response and speech prostheses.

Minimal Recursion Semantics is not a theory of semantics but rather a system of semantic representations. As such, in order to develop a grammar of a particular language, there are any number of design decisions that must be made about the content of the semantic representations. In this paper, we present some design decisions that have emerged in work on two broad-coverage grammars the English Resource Grammar (ERG: Flickinger 2000) and the JaCY Japanese grammar (JaCY: Siegel and Bender 2002). While the influences on these design decisions are myriad, perhaps the most important guiding principle is the following: The semantic representations produced should include all grammatically relevant distinctions, while remaining as concise as possible. Thus, for example, in the Matrix as in the ERG, past tense is represented simply as the attribute-value pair [TENSE past] (on an event variable) rather than a more elaborate Reichenbachian representation relating the event time to the time of utterance, because this more elaborate representation can be unambiguously derived from the more parsimonious one given. (For more discussion, see §5 below and §6 of Copestake et al. 2003.) We hope that the descriptions in the following sections of the design decisions taken so far will provide guidance for grammar engineers confronting new linguistic phenomena.

3 Basic Semantic Objects

This section provides an overview of the semantic objects defined in the Matrix, used in the specification of the values of CONT in signs, both words and phrases. Figure 1 gives a portion of the type hierarchies under *avm* and *sort*, including all of the basic semantic objects.

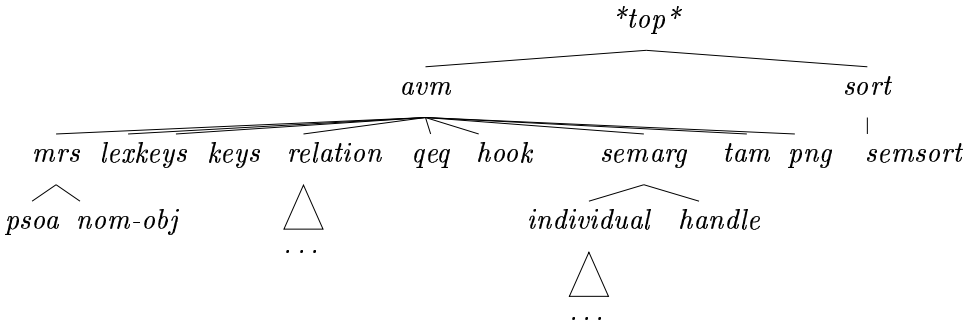


Figure 1: Type Hierarchy of Basic Semantic Objects

3.1 *mrs*

The flat semantic representations assigned to each word or phrase in MRS comprise three components:

1. RELS - a bag of atomic predications, each with a 'handle' (used to express scope relations) and one or more roles;

2. HCONS - a set of handle constraints which reflect syntactic limitations on possible scope relations among the atomic predications;

3. HOOK - a group of distinguished externally visible attributes of the atomic predications in RELS, used in combining the semantics of this sign with the semantics of other signs.

Thus objects of type *mrs* (i.e., the value of the features CONT(ENT) and C(ONSTRUCTIONAL)-CONT of *sign*s in the Matrix) are constrained as in (1).

(1) *mrs*: $\begin{bmatrix} \text{HOOK} & \textit{hook} \\ \text{RELS} & \textit{diff-list} \\ \text{HCONS} & \textit{diff-list} \\ \text{MSG} & \textit{basic_message} \end{bmatrix}$

The value of RELS is a difference list[1] of *relation*s (§3.2) and the value of HCONS is a difference list of *qeq*s (§3.3). The value of HOOK is a feature structure of type *hook* (§3.4). Finally, the value of MSG is a feature structure of type *basic_message*, used to constrain the propositional type of a sign (§6.6.1.

The type *mrs* also has two subtypes *nom-obj* and *psoa*, following Pollard and Sag 1994, corresponding to semantic representations of nominal signs (with a distinguished *ref-ind*) and predicative signs (with a distinguished *event*). The types *ref-ind* and *event* are described in §3.5 below. This distinction is encoded in the following two constraints:[2]

(2) *psoa*: $\left[\text{HOOK} \quad \left[\text{INDEX} \quad \textit{event}\right]\right]$

(3) *nom-obj*: $\left[\text{HOOK} \quad \left[\text{INDEX} \quad \textit{index}\right]\right]$

3.2 *relation*

The heart of an *mrs* is a bag of elementary predications, implemented here as a difference list of objects of type *relation* – i.e., the value of the feature RELS.[3] All relations bear values for the three features introduced on the type *relation*, as shown in (4):

[1] These lists are implemented as difference list to allow for appending without relational constraints.

[2] On *hook*, see §3.4 below.

[3] Note that the Matrix doesn't include typed difference lists, so that the value of RELS is in fact only constrained to be a difference list without further restriction – it could just as well be a difference list of *synsem*s. By convention, and in order to produce well-formed semantic representations, however, it is always a difference list of *relation*s.

(4) $$\textit{relation}: \begin{bmatrix} \textsc{lbl} & \textit{handle} \\ \textsc{pred} & \textit{string} \\ \textsc{wlink} & \textit{list} \end{bmatrix}$$

3.2.1 LBL

The value of LBL is a *handle*, which is used to express scope relations. The LBL value of one relation may be identified with (i) the LBL value of one or more other relations, (ii) a role within some other relation, (iii) a value within a handle constraint (*qeq* – see §3.3 below), and/or the value of LTOP in a *hook* (see §3.4 below). Relations sharing a LBL value are interpreted as conjoined. For example, in *The big dog slept.*, the relations introduced by *big* and *dog* will share a LBL value. *Qeq* constraints and direct identification of LBL values with argument roles in other relations are used to express scopal interactions among relations. For the most part, this is achieved with *qeq* constraints, which identify semantic argument positions where quantifiers can intervene scopally when underspecified *mrs*s are resolved to fully scoped representations. For those handle argument positions where quantifiers cannot intercede, direct reentrancy between a LBL value and a role is employed. For an example, see §6.6.1 below. Note that we will often refer to the LBL value of a relation as its handle.

3.2.2 PRED

The value of PRED is a string, which serves to distinguish particular relations. Earlier versions of the ERG included a separate type for each distinct relation, leading to a very flat (and very large) type hierarchy. We have since found it preferable to distinguish relations via the string-valued PRED feature, and to reserve the subtypes of *relation* (see §4) for those types that introduce features. This change also opens the door to more interesting re-entrancies of PRED values, explored in the ERG for the semantics of degree specifiers, and perhaps also useful for some treatments of coordination or gapping.

For compatibility with RMRS ((Copestake 2003)) and software designed to integrate deep and shallow processing, PRED values should conform to the following templates:

(5)
`_orth_pos_sense_rel`	(lexically introduced predicates)
`sense_rel`	(abstract predicates introduced by constructions)

The `orth` component is a string corresponding to the (stem) orthography of the lexical entry, at least for all open-class words, and typically also for closed-class words. By using the stem orthography, we enusre that predicate names used in *mrs*s produced by deep grammars will be interoperable with predicate names used in *mrs*s produced by robust shallow processors, which name the predicates based on (lemmatized) forms in the input. The leading underscore is used to distinguish predicate names introduced by specific lexical entries from those introduced by constructions or by lexical types supplying a common predicate for a class of lexical entries.

The `pos` component is one of a closed set of single lowercase letters interpreted as follows:

(6)

n	noun
v	verb
j	adjective
r	adverb
p	preposition
q	determiner (quantifier)
m	message
x	all other closed-class predicates

We use this POS-based information (such as might be accessible from a POS-tagger) for coarse-grained sense distinctions. Finer-grained distinctions can be made (in a precision grammar) via the `sense` component. The `sense` component can consist of any sequence of characters (letters, numbers, etc.), excluding the underscore which is used to separate the components of the name. In the ERG, verb particle constructions are handled semantically by having the verb contribute a relation particular to that combination. We distinguish these relations by placing the particle's orthography in the `sense` field. Unlike the other components, the `sense` component is optional, and if omitted, its separating underscore is also omitted. By convention, a predicate name with no sense component is interpreted as underspecified for sense, so if more than one sense is present in the lexicon for a given orthography and part of speech, each of these predicate names should have a sense component.

Every relation and predicate name ends in **_rel**, for the convenience of the grammar writer, particularly to avoid possible namespace collisions. This suffix (and the leading underscore) can of course be suppressed by MRS display methods if desired.

So for example, the following predicate names are correct for the corresponding words:

(7)

aardvark	**_aardvark_n_rel**
bank	**_bank_n_2_rel**
bank	**_bank_v_turn_rel**
look	**_look_v_up_rel**

Finally, one further detail of formatting should be mentioned: Words with single lexical entries whose orthography is conventionally spelled with a space, such as the English use of *ad hoc*, appear with the whole orthography in the `orth` component, but with the space(s) replaced by the plus sign. So the following example is also correct:

(8) *ad hoc* **_ad+hoc_j_rel**

3.2.3 WLINK

The feature WLINK serves to link the relation to an element in the input string, so that applications can reconstruct the input string source of a given relation in the corresponding semantics. Since this feature does not interact with the rest of the semantic composition machinery in the Matrix, we will omit it from our AVM descriptions in the rest of this

paper. It is included as part of the Matrix because it provides a useful experimental link later on, when developing applications that use a Matrix-derived grammar. The grammar engineer can safely ignore this feature during grammar development. If an application seesm to require it, contact the authors for details on how to put it to use.

3.3 *qeq*

As mentioned above, scopal relations in MRS are represented via handles, which appear as the value of the feature LBL and also as the value of certain roles within scopal relations (e.g., the RSTR value of quantifiers – see §4.2.1). However, for many applications, including machine translation, fully specified scope relations are not required. Furthermore, any surface string with multiple noun phrases (and therefore multiple quantifiers, overt or implicit) is going to be ambiguous with respect to scopal relations. Rather than build separate parses for each scoping, and potentially have to choose between them in a given application, it is preferable to leave scopal relations underspecified, to the extent that the grammar doesn't in fact constrain them.[4] In Matrix-derived grammars, following MRS specifications, this is achieved as follows:

1. The BODY (i.e., scope) of all quantifiers is left unconstrained.
2. Most other handle-taking argument positions are not directly linked to the handle of some relation.
3. Rather, the two are related via a *qeq* ('equality modulo quantifiers') constraint.

Thus the BODY value of quantifier relations can be resolved in such a way that the quantifiers 'float in' wherever there's a 'space' left by a *qeq* constraint.

In the implementation, these handle constraints (restrictions on scopal interactions) are represented via the feature HCONS. The value of HCONS is a difference list (again, representing a bag) of *qeq*s.[5] *Qeq*s, in turn, are constrained as follows:

(9) *qeq*: $\begin{bmatrix} \text{HARG} & \textit{handle} \\ \text{LARG} & \textit{handle} \end{bmatrix}$

The HARG is identified with the handle-taking argument position and the LARG is identified with the LBL (handle) of the outscoped relation. Examples of parts of the grammar that impose such constraints are given in §6.6 below.

While the MRS specification in Copestake et al. 2003 leaves open the possibility of different kinds of handle constraints, only *qeq* constraints have proven necessary so far for wide-coverage grammars of English (ERG) and Japanese (JaCY), and so only *qeq* constraints are currently implemented in the Matrix.[6]

[4] For discussion of underspecification of scope, see Copestake et al. 2003 and references given there.

[5] Once again, the difference list is not typed. See note 3.

[6] However, a large grammar implementation for German (described in Wahlster and Karger 2000) makes crucial use of *leq* (less than or equal) scopal constraints, and recent work on extending MRS representations to robust processing (((Copestake 2003)) also employs *leq* constraints.

3.4 *hook*

Where the values of RELS and QEQ give a rich representation of the semantics of any given sign (word or phrase), more information is needed in order to be able to combine the semantic representations of signs in order to compositionally build semantic representations of larger phrases. For example, consider the partial feature structures for the *mrs*s produced by the ERG for the signs *the dog* and *barks* in (10):[7]

(10) a.
$$\begin{bmatrix} \text{RELS} & \langle! \begin{bmatrix} \text{LBL} & \textit{handle} \\ \text{PRED} & \textbf{_the_q_rel} \\ \text{ARG0} & \boxed{2} \\ \text{RSTR} & \boxed{3} \\ \text{BODY} & \textit{handle} \end{bmatrix}, \begin{bmatrix} \text{LBL} & \boxed{5} \\ \text{PRED} & \textbf{_dog_n_rel} \\ \text{ARG0} & \boxed{2} \end{bmatrix} !\rangle \\ \text{HCONS} & \langle! \begin{bmatrix} \textit{qeq} & \\ \text{HARG} & \boxed{3} \\ \text{LARG} & \boxed{5} \end{bmatrix} !\rangle \end{bmatrix}$$

b.
$$\begin{bmatrix} \text{RELS} & \langle! \begin{bmatrix} \text{LBL} & \textit{handle} \\ \text{PRED} & \textbf{_bark_v_rel} \\ \text{ARG0} & \textit{event} \\ \text{ARG1} & \textit{semarg} \end{bmatrix} !\rangle \\ \text{HCONS} & \langle! \; !\rangle \end{bmatrix}$$

In composing the *mrs* for the larger phrase *the dog barks*, we would like to identify the ARG0 of the **_dog_n_rel** with the ARG1 of the **_bark_v_rel**. But how can the relevant parts of the grammar (in this case, the lexical entry for *bark*) gain access to the right value of the right relation? While the value of RELS is implemented as a difference list, it is notionally to be treated as a bag, so it would be unprincipled to make reference to the position of a relation in the list. In fact, it would also be impractical for a grammar of any interesting size: the list position of the relation contributed by the head noun in a noun phrase is affected by what else there is in the NP. For example, in *the big dog*, the **_big_j_rel** would intervene (in this implementation) between the **_the_q_rel** and the **_dog_n_rel**. The solution is to use the value of the attribute HOOK to 'publish' or make visible externally just those elements of the *mrs* that the grammar will need access to for semantic composition.

Thus the value of HOOK represents a hypothesis about which information may be accessed externally. The Matrix provides for three compositionally relevant properties of a sign's semantics, encoded as features for objects of type *hook*:

(11)
$$\textit{hook}: \begin{bmatrix} \text{LTOP} & \textit{handle} \\ \text{INDEX} & \textit{individual} \\ \text{XARG} & \textit{individual} \end{bmatrix}$$

[7]Following LKB conventions, difference lists are represented with the brackets ⟨! !⟩, consistent with the abbreviatory conventions employed in the TDL ((Krieger and Schaefer 1994)) syntax adopted in the LKB.

The value of LTOP is the local top handle, the handle of the relation(s) with the widest scope within the constituent, modulo quantifiers. This attribute is accessed by semantic heads in phrasal constructions in order to impose further scopal constraints involving that handle when composing the semantics of the phrase.

The value of INDEX is the distinguished non-handle variable supplied by the sign, identified with the INDEX of the semantic head daughter, and usually the ARG0 of the main relation introduced by the syntactic head of the constituent. If the *mrs* is a *nom-obj*, this will be a *ref-ind* (referential index). If the *mrs* is a *psoa*, this will be an *event*. (See §3.1 above on *psoa* and *nom-obj* and §3.5 below on *ref-ind* and *event*.) This information is accessed by semantic heads in order to identify indices (including event indices) with (non-scopal) argument positions in predications: e.g., the ARG1 (barker) of the **_bark_v_rel** or the ARG1 (modified event) of the relation for an intersective adverb like *happily*.

Finally, the value of XARG (mnemonic for 'external argument') is the index of the single argument in a phrase (in accusative languages, typically the subject) which can be controlled. This information will be accessed by semantic heads in raising and control constructions, open adjuncts, and constructions like English tag questions. See §6.7 for further exemplification.

3.5 *semarg*, *tam*, and *png*

The values of LBL, HARG, LARG, and all role features (e.g., ARG0 etc., see §4) are objects of (subtypes of) type *semarg*. This type introduces the non-linguistic feature INSTLOC, which is used for skolemization of variables in generation and whose value should never be further constrained by the grammar. The hierarchy below *semarg* is shown in Figure 2.

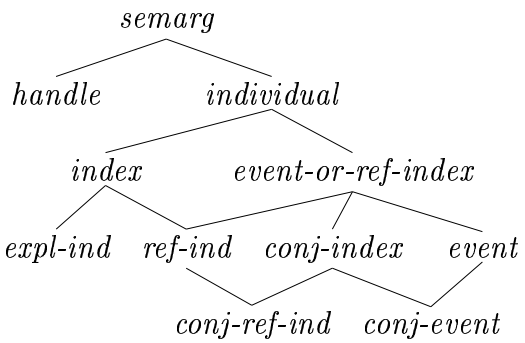


Figure 2: Type hierarchy rooted in *semarg*

There are two subtypes of *semarg*. The first, *handle*, is the type of the value of all handle-taking features (LBL, HARG, LARG, MARG, RSTR, BODY), and can be the value type for a semantic role filling one of the features ARG1, ARG2, ... There are no subtypes of *handle*, nor is it anticipated that grammar engineers will need to add any in the course of grammar development. Furthermore, there are no features introduced on the type *handle* (although it inherits INSTLOC from *semarg*).

The other immediate subtype of *semarg* is *individual*, which includes expletive indices (*expl-ind*), event indices (or event variables) *event* and ordinary referential indices (*ref-ind*). In addition, the Matrix provides for coordinate indices (*conj-ind* with subtypes *conj-event* and *conj-ref-ind*) which are introduced by coordination constructions (see §4.2.3). Note that *conj-event* and *conj-ref-ind* inherit from *event* and *ref-ind*, respectively. This ensures that they are compatible with any environment that requires an *event* or *ref-ind*, and, conversely, that it is not possible to specifically select a non-coordinated event or referential index.[8]

The contrast between *expl-ind* and *event-or-ref-index* is used to constrain the distribution of expletive NPs (e.g., English *it* and *there*, German *es*, or French *il*).[9] Such expletive pronouns are given an INDEX value of type *expl-ind* (which is exceptionally not linked to any argument position in their main relation, since they don't introduce any relation). Ordinary selecting heads can then require dependents with contenful INDEX values (i.e., *event-or-ref-index*s, or some subtype thereof). In languages like English (and perhaps Dutch) with multiple expletive pronouns, subtypes of *expl-ind* can be added to distinguish the different pronouns. In the ERG, we have found this technique useful not only for implementing the selection of particular expletive pronouns by particular heads (e.g., *rains* vs. existential *be*), but also for pronoun matching in tag questions (see Bender and Flickinger 1999).

The type *individual* introduces the feature SORT, which can be used to identify the grammatically relevant semantic sort (if any) of the sign, for use in semantic selection of this sign as a dependent in a larger phrase. The value of this attribute is of type *semsort*, discussed in §3.7.

The types *event* and *ref-ind* each introduce one feature, as shown in (12):

(12) a. *event*: $\left[\text{TAM} \quad \textit{tam}\right]$

b. *ref-ind*: $\left[\text{PNG} \quad \textit{png}\right]$

The features TAM and PNG encode tense-aspect-mood information (properties of events) and agreement information (person, number, and gender – properties of referential indices; cf. Pollard and Sag 1994), respectively. In the current version of the Matrix (v 0.6), the type *tam* (unsurprisingly) introduces the features TENSE, ASPECT, and MOOD. This may be too strong a constraint, since languages may well conflate two or more of these properties; see the discussion of the *png* type immediately following. We will have more to say about the analyses of agreement in §5 below.

(13) $\textit{tam}: \begin{bmatrix} \text{TENSE} & \textit{tense} \\ \text{ASPECT} & \textit{aspect} \\ \text{MOOD} & \textit{mood} \end{bmatrix}$

[8]The hierarchy could of course be extended to allow such selection, but we expect that it will not be required.

[9]Although in general we intend the types given in the Matrix proper (as opposed to separate modules that will eventually come with it) to be largely language-independent, *expl-ind* is probably not useful in languages which allow pro-drop to the extent that Japanese does, and which therefore have no use for expletive elements.

The type *png*, on the other hand, introduces no features, though it is intended to provide the locus for constraints on (semantic) person, number, and gender. These three dimensions are not necessarily distinct for a given language, as seen in English where person and number are usually conflated morphologically, motivating a basic distinction between third-singular and non-third-singular inflectional types, with further subtypes for non-third-singular.[10] Thus in English the most natural feature structure for the *png* type is to have a merged person-number PN attribute along with a gender GEN attribute. Clearly, in other languages separate attributes for person and number are well motivated, leading us to leave the elaboration of the *png* type as language-specific for now.

3.6 *semsort*

The type *semsort* is the root of a hierarchy of sorts which serves to represent grammaticized semantic distinctions which are used in the selection of dependents (subjects, complements, specifiers, and modifiers). We expect it to include a small hierarchy of grammatically salient semantic types such as *animate* or *time*, as well as more specific types to support selection of closed-class items, such as preposition selection by verbs or auxiliary matching in English tag questions (Bender and Flickinger 1999). Note that *semsort* is made a subtype of *sort* to make explicit the claim that these types used for semantic selection do not themselves introduce attributes of their own.

The ERG illustrates several uses of semantic selection, which motivate some particular subtypes of *semsort*. These sorts in the ERG include at least the human/nonhuman distinction reflected in the choice of relative pronoun (*the person *which/whom I met*), and the various semantic subtypes introduced by prepositions (temporal, locative, directional, stative, etc.). Verb-preposition dependencies in English, for example, can be encoded by having the verb constrain the KEY value of its PP complement. In some cases, the selecting verb might not be looking for a particular preposition's KEY value, but rather impose a more abstract constraint on a complement's key, such as the PP complement for the English verb *put*, whose KEY value might be constrained to those introduced by a subclass of locative prepositions, to preclude analyzing e.g. **Kim put the chair for Sandy.* Thus both the leaves and the intermediate types in the *semsort* hierarchy can be useful.

The subhierarchy under *semsort* will be language-specific, reflecting grammaticized semantic properties motivated by constructions in that language. The analysis of Japanese numeral classifiers, for example, can be implemented using these semantic sorts, as can the quite idiosyncratic constraints for English on preposition selection with temporal nouns. Here, for example, noun phrases like *Tuesday* or *the fifteenth* (denoting days of the week or days of the month) introduce the semantic sort *day* which is selected by (the relevant lexical entry for) the preposition *on* but not *in*, to admit *Kim arrived on Tuesday* but not **Kim arrived in Tueaday.* Such idiosyncratic collocational constraints often reflect natural semantic distinctions, but are not predictable cross-linguistically.

Note that, as strings, PRED values are not organized into a hierarchy. Furthermore expect to only need to identify particular relations (via *semsort*s which correspond to specific PRED values) in a reasonably small number of closed classes of lexical items. We therefore see now

[10] Cf. Flickinger 2000 for discussion of this part of the type hierarchy.

value in providing an external pointer to the PRED value of the main predication contributed by the lexical head (syntactic or semantic) of a phrase.

The final attribute of objects of type *mrs* is MSG. This attribute has as its value a subtype of the type *message* or the type *no-msg*, depending on whether the phrase is clausal or non-clausal. *Message*s are our representation of the semantic types of clauses and are described in more detail in §4.2.2 below. For clausal phrases, the value of MESSAGE will be a pointer to the relevant message relation in the RELS list of that phrase.

3.7 *keys*

The next type of object to consider is *keys*, not exactly a semantic object, but one useful for the interface between syntax and semantics. This type is introduced to serve as the value of the (syntactic) HEAD feature KEYS. The purpose of KEYS is to provide constraints for semantic selection of dependents (complements, specifiers, subjects, and modifiers). The attributes in *keys* include two called KEY and ALTKEY, which can be used by selecting heads or by constructions to constrain a dependent phrase semantically. Since the KEYS attribute is a HEAD feature, the grammar ensures that these semantic properties of a phrase propagate up the syntactic head path.

The type *keys* is constrained as shown in (14):

(14) $\begin{bmatrix} \text{KEY} & \textit{predsort} \\ \text{ALTKEY} & \textit{predsort} \end{bmatrix}$

The feature KEY provides a constraint on the *predsort* of a phrase, which can be but is not necessarily identified with the PRED value of one of the relations in the RELS list of that phrase. Since generalizations about semantic selection may make reference to more than one such semantic property of a sign, the Matrix provides a second KEYS attribute called ALTKEY. The idea here is that a modifier, for example, may constrain the phrase it modifies along one dimension of semantic selection, while a verb taking that same modified phrase as a complement may need to constrain it along a second dimension.

Note that as a head feature, the value of KEYS on the mother node of a headed-phrase will be identified with the KEYS value of the head daughter. For non-headed phrases such as coordinate constructions, each such construction type will have to stipulate the values for the attributes in KEYS on the mother node. These values may come from one or the other of the daughters, or may be supplied directly by the construction itself.

3.8 *lexkeys*

The last type of object to consider here is the type *lexkeys*, which is defined in the Matrix to provide attributes which are not linguisticslly significant, but which provide some convenient shorthand notation for the grammar writer when defining the lexical type hierarchy. In particular, this type introduces two attributes that simplify the expression of constraints on relations introduced by a lexical type, and two attributes that point to the KEY attribute of complements of the lexical type.. The grammar writer can decide whether or not to make use of these shorthand attributes (the latter two bearing a leading double dash as a reminder

that they are only abbreviations for longer path names). The type *lexkeys* is the value of the LOCAL feature LKEYS, and nothing in the Matrix propagates the value of this feature up to phrases. Thus, the features on *lexkeys* are only used (if at all) in simplifying the definitions of lexical entries.

The type *lexkeys* is constrained as follows:

(15)
$$\textit{lexkeys}: \begin{bmatrix} \text{KEYREL} & \textit{relation} \\ \text{ALTKEYREL} & \textit{relation} \\ \text{--COMPKEY} & \textit{semsort} \\ \text{--OCOMPKEY} & \textit{semsort} \end{bmatrix}$$

The two features KEYREL and ALTKEYREL are available to provide pointers to each of two relations introduced by a lexical entry, for easier definitions of constraints on values of those relations within the lexical type hierarchy. Typically, the value of KEYREL will be a pointer to the relation in the RELS list of a lexical entry which introduces its INDEX value as the ARG0 of that relation. The value of ALTKEYREL will be unbound for most lexical entries, since most introduce a single relation in the RELS list, but for an entry which introduces more than one relation, this attribute provides a convenient pointer to a second relation on the RELS list.

The features --COMPKEY and --OCOMPKEY provide pointers to the KEY values of two complements of a lexical entry. The relationship between --COMPKEY/--OCOMPKEY and KEYS.KEY of the relevant complements will be established in lexical types.[11] This allows specific lexical entries to do semantic selection of complements by simply constraining their own --COMPKEY and/or --OCOMPKEY values, as in the following (partial) lexical entries adapted from the ERG for the verb *abstain* and the empty preposition *from* as in *Kim abstained from the vote*:

(16)
$$\textit{abstain_v1}: \begin{bmatrix} \textit{v_empty_prep_intrans_le} & \\ \text{STEM} & \langle \text{ “abstain” } \rangle \\ \text{SYNSEM} & \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{LKEYS} & \begin{bmatrix} \text{KEYREL} & \textbf{_abstain_v_from_rel} \\ \text{--COMPKEY} & \textbf{_from_p_sel_rel} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(17)
$$\textit{from_prtcl}: \begin{bmatrix} \textit{p_prtcl_le} & \\ \text{STEM} & \langle \text{ “from” } \rangle \\ \text{SYNSEM} & \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{LKEYS} & \begin{bmatrix} \text{KEYREL} & \begin{bmatrix} \text{PRED} & \textbf{_from_p_sel_rel} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The lexical type for the verb *abstain* includes as part of its definition the following reentrancy which enables the use of this –COMPKEY shortcut:

[11] In the case of the entry for *abstain* given here, the relevant constraint in the ERG is on the type *unsat_two_arg_subst*, which is a supertype of the SYNSEM value of *v_empty_prep_intrans_le*.

(18)
$$\begin{bmatrix} \textit{v_empty_prep_intrans_le} \\ \textsc{synsem} \begin{bmatrix} \textsc{loc} \begin{bmatrix} \textsc{cat} \begin{bmatrix} \textsc{val} \begin{bmatrix} \textsc{comps} \begin{bmatrix} \textsc{first} \begin{bmatrix} \textsc{loc} \begin{bmatrix} \textsc{cat} \begin{bmatrix} \textsc{head} \begin{bmatrix} \textsc{keys} \begin{bmatrix} \textsc{key} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \textsc{lkeys} \begin{bmatrix} \textsc{--compkey} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The lexical type for semantically empty prepositions includes the following reentrancy which shows how the constraints in the two lexical entries above will interact correctly, without making any reference to the attribute LKEYS during processing, as desired.

(19)
$$\begin{bmatrix} \textit{p_prtcl_le} \\ \textsc{synsem} \begin{bmatrix} \textsc{loc} \begin{bmatrix} \textsc{cat} \begin{bmatrix} \textsc{head} \begin{bmatrix} \textsc{keys} \begin{bmatrix} \textsc{key} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \textsc{lkeys} \begin{bmatrix} \textsc{keyrel} \begin{bmatrix} \textsc{pred} & \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

3.9 Summary

This concludes our tour of the basic semantic objects defined in the Matrix. In this section we have briefly touched on some aspects of the linguistic analyses that motivate each type of object. The following sections flesh out these linguistic analyses in more detail.

4 Relations and Argument Features

This section describes the various subtypes of *relation* posited in the Matrix and provides guidelines for introducing new subtypes of *relation*. The top of the type hierarchy below *relation* is shown in Figure 3.

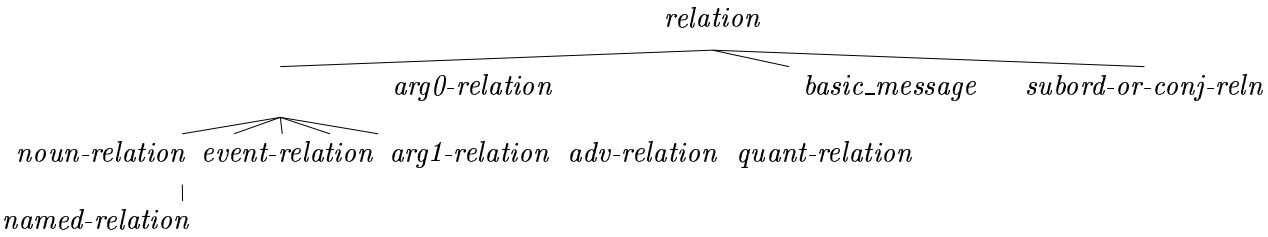


Figure 3: Partial type hierarchy below *relation*

In the following subsections we will describe ordinary predicates (i.e., the subtypes of *arg0-relation* introduced by nouns, verbs, prepositions, adjectives and adverbs – §4.1), and the special relation types defined for quantifiers (§4.2.1), messages (§4.2.2), and subordination and coordination (§4.2.3).

While we expect that grammar development for particular languages will require the addition of some abstract relation types to the set proposed here, we recommend not positing a type for each lexical relation, but rather using the feature PRED (see §3.2 above) to distinguish different lexical relations of the same type. New relation types are merited only in two circumstances: (i) when a feature needs to be introduced that is relevant for some relations but not others; or (ii) when the values of one or more attributes of an existing relation are consistently constrained in the same way across multiple contexts in the grammar.

4.1 Ordinary Predicates

4.1.1 *arg0-relation* subtypes

The vast majority of lexical entries introduce only a single relation, and furthermore one that is an instance of *arg0-relation* or one of its subtypes, including *noun-relation*, *event-relation*, and *adv-relation*. The type *arg0-relation* is constrained as follows:

(20) *arg0-relation*: $\left[\text{ARG0} \quad \textit{individual}\right]$

Thus all open-class lexical items introduce a relation with the ARG0 role. The value of this role is an *individual*. For nouns, verbs, and adjectives, the ARG0 of the main relation (i.e., the KEYREL value) should be identified with the HOOK.INDEX, and this identification should be done on a general supertype.

For ordinary nouns, the value of ARG0 will be a (referential) index which serves as a pointer to the entity referred to by the NP. This same index is also the value of the ARG1 feature in any relations introduced by intersective modifiers of the head noun, and also the value of a role feature in the relation of any lexical predicate selecting the NP as a semantic argument. The type *noun-relation* is constrained appropriately:

(21) *noun-relation*: $\left[\text{ARG0} \quad \textit{ref-ind}\right]$

As discussed in §3.5 above, objects of type *ref-ind* bear the feature PNG. This means that the agreement information associated with a noun will also be found in LKEYS.KEYREL.ARG0, but this information doesn't propagate up to the phrase, and so agreement information should always be accessed through the CONT.HOOK.INDEX.

For verbs, which introduce relations of type *event-relation*, the value of ARG0 is the event variable identified as the INDEX value. Again, this *event* will also show up as the value of a role feature in modifier relations such as those expressed by (intersective) adverbs and by PPs modifying verbal phrases. Other elements that might appear to notionally take events as arguments (e.g., clause-embedding verbs or scopal adverbs like *probably*) actually take the handle of the phrase as their argument. The type *event-relation* is therefore constrained as follows:

(22) *event-relation*: $\left[\text{ARG0} \quad \textit{event}\right]$

As discussed in §3.5 above, objects of type *event* bear the feature TAM. This means that tense, aspect and mood information associated with an event are encoded in the ARG0 of the relation describing that event. Again, as there is no pointer to the whole relation that gets passed up to the phrasal level, this information and the event index itself should be accessed via the path HOOK.INDEX for semantic selection or composition.

The Matrix provides one particularly useful subtype of *noun-relation* called *named-relation*:

(23) *named-relation*: $\begin{bmatrix} \text{PRED} & \textbf{named_rel} \\ \text{CARG} & \textit{string} \end{bmatrix}$

This type is used for proper names (including names of months, days of the week and days of the month), and it introduces a feature CARG ('constant argument') which takes as its value a string representing the name of the named entity. It further constraints the PRED value to be **named_rel**. Thus all proper nouns contribute relations of the same type and with the same pred value. Their contribution to the semantics is distinguished solely by their CARG (and of course their ARG0s, which will be distinct for each proper name in a single sentence).

Note that as *named-relation* is a subtype of *noun-relation*, it will introduce a referential index (*ref-ind*) which must be bound by some quantifier in order to form a well-formed *mrs*. In the ERG, this is achieved by means of a non-branching rule which adds the quantifier to a proper noun, which usually but not always lacks an explicit determiner. Given noun phrases in English like *the younger Smith* and *some Roberts*, the ERG defines lexical entries for proper names as syntactically nouns, not NPs, and employs a specialized unary syntactic rule to construct a noun phrase from an N-bar headed by a proper noun (and usually consisting only of that noun). The Matrix will support this analysis of proper names, but will of course also enable the construction of grammars for languages where proper names are simply lexical NPs, in which case each lexical entry's semantics will consist of two relations: a *named-relation* and a *quant-relation* to bind its referential index.

4.1.2 *arg1-relation*, *arg12-relation*, ...

As discussed in §4.1.1, the value of ARG0 is the distinguished *ref-ind* or *event* of a relation. Many relations, of course, take further arguments. These are represented with the features ARG1, ARG2, ARG3 and ARG4. It is important to note that there is no independent interpretation of thematic roles attached to these feature names. That is, ARG1 cannot be taken as equivalent to something like AGENT wherever it is used. Rather, we understand the precise interpretation of the role names to be dependent on the relation they appear in. This interpretation is to be specified in a separate component of the grammar called the Sem-I ('semantic interface'), which provides such information as is needed to map from *mrs*s to application-specific representations. The Sem-I is further described in §8 below.

Figure 4 shows the type hierarchy below *arg0-relation*, filling in the information below *event-relation*, *noun-relation*, and *arg1-relation*, which was abbreviated in Figure 3 above. The features ARG1 through ARG4 are introduced by the types *arg1-relation* through *arg1234-relation*, as shown in (24). The values of the argument features on these types are only

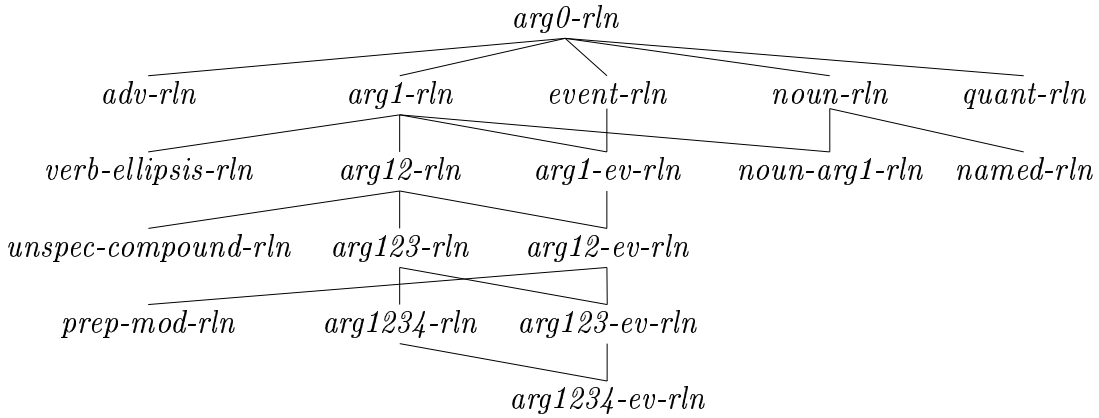


Figure 4: Hierarchy below *arg0-relation*

contrained to be *semarg*s, as for any given predicate they may be *individual*s (*ref-ind*s or *event*s), or *handle*s in the case of scopal predicates.

(24) a. *arg1-relation*: $\left[\text{ARG1} \quad \textit{semarg}\right]$

b. *arg12-relation*: $\left[\text{ARG2} \quad \textit{semarg}\right]$

c. *arg123-relation*: $\left[\text{ARG3} \quad \textit{semarg}\right]$

d. *arg1234-relation*: $\left[\text{ARG4} \quad \textit{semarg}\right]$

Note that, in keeping with the strategy for interpretation outlined above, it is not possible to have an ARG3 without an ARG1 and an ARG2. That is, if a relation takes two arguments beyond its distinguished event/index (ARG0), they will always be labelled ARG1 and ARG2. Furthermore, the ARG1 should always correspond to the first (least oblique) syntactic argument, the ARG2 to the second (next least oblique) syntactic argument, and so on (see §6.4 and Flickinger and Bender forthcoming).

The type *arg1-relation* does not inherit from *event-relation*, to allow for the possibility of relational nouns and other relations that take more than one argument but do not express events. The types *arg1-ev-relation* through *arg1234-ev-relation* provide event relations with one to four arguments. Thus (semantically) intransitive verbs introduce *arg1-ev-relation*s, (semantically) transitive verbs introduce *arg12-ev-relation*s, etc. Ordinary nouns introduce *noun-relation*s, which only have an *arg0*. Relational nouns (such as *picture*) introduce a relation *noun-arg1-relation* which inherits from both *noun-relation* and *arg1-relation* (not *arg1-ev-relation*). A noun with two semantic arguments would require a relation that inherits from *noun-relation* and *arg12-relation*, and similarly for a noun with three arguments, if such exist (apart from deverbal nouns, which are treated for English via lexical rule in the ERG). These types all inherit from *arg0-relation* and through it *relation*, so they will all also bear the features ARG0, LBL, PRED, and WLINK.

Note that verbs are not the only lexical entries which introduce subtypes of *event-relation* in their semantics. Adjectives can in some languages serve directly as predicates, be subject to tense and aspect constraints, and take one or more arguments, all analogous to verbs. Likewise, prepositions can also appear as the heads of predicative phrases, so they will also introduce subtypes of *event-relation* in their semantics. It is worth noting that the choice of the name *event-relation* is intended to include states as well as processes, as required by verbs as well as adjectives and prepositions. At present the Matrix does not introduce events into the relations for nouns, though it might be argued that examples in English like *the current president* motivate treating at least some nouns as event-bearing. Since we do not know of grammar-internal constraints that require making reference to events on nouns, we will expect such temporally constrained noun phrases to be interpreted outside of the grammar.

Like verbs, adjectives and prepositions identify their HOOK.INDEX value with the ARG0 value in their main relation, enabling them to appear as heads of predicative phrases, supplying an event which can be constrained for tense and aspect. In the ERG, for example, the copula *be* does not supply its own relation, but only constrains tense and aspect since it often combines with a verb participle that already supplies the event relation, as in *Kim was leaving*. Since that grammar uses the same copula *be* for *Kim was angry*, the adjective *angry* must supply an event relation whose ARG0 event will be constrained in this example to past tense. Now since adjectives and prepositions can head phrases that modify nouns, they must also expose the argument position they will identify with the index of the phrase they modify. This "external argument" of a PP or AP is its HOOK.XARG value, identified in the lexical head's feature structure with the value of the rather long path HEAD.MOD.FIRST.LOCAL.CONT.HOOK.INDEX. This value will be unified by the grammar rules for modification with the semantic index of the noun being modified. These two HOOK attributes INDEX and XARG enable the semantic composition desired for intersective modification of nouns by adjectives or PPs, and of verbs by PPs, as well as the use of adjectives and prepositions in predicative constructions. The machinery should extend to more interesting examples like *The dog currently in the park is barking*, where *currently* temporally constrains the event introduced by the PP *in the park* (its HOOK.INDEX) while the PP's HOOK.XARG is identified with the HOOK.INDEX of the noun *dog*.

Adverbs, unlike adjectives and prepositions, do not appear predicatively, and hence do not need to introduce an independent event. So they simply identify the value of their main relation's ARG0 with the index of the phrase they modify, and further identify this value with their own HOOK.INDEX.

4.2 Special Relations

The Matrix also provides a small number of additional semantic relations which introduce additional attributes for easier readability for both grammar writers and application developers. These special relations could of course be defined just using the above inventory of relation attributes (PRED and ARG1 ... ARGN), since the Sem-I would provide an unambigous interpretation of these features within a given relation type. But it has proven to be convenient in practice to supply the following relations with their own particular attributes.

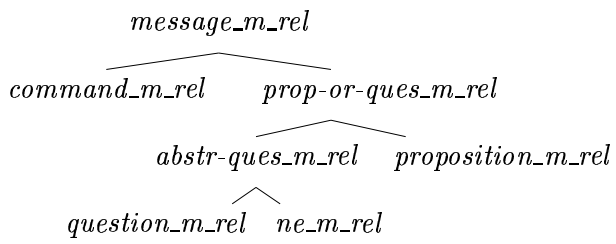


Figure 5: Subhierarchy under *message*

4.2.1 Quantifiers

The lexical types for quantifiers like *some* and *every* introduce a *quant-relation* which is also a subtype of *arg0-relation*, but thie type introduces two additional features, RSTR (restrictor) and BODY. As scopal features, they both take values of type *handle*. The RSTR will be related to the top handle of the quantifier's restriction (i.e., the N′ that takes the quantifier as a specifier) via a *qeq* constraint (see §6.6.2 below for details on how this constraint is introduced). The BODY is left unbound: this is what allows quantifiers to have varied scoping possibilities (see Copestake et al. 2003). The value of ARG0 is the referential index that the quantifier binds (see §6.5).

The type *quant-relation* is thus constrained as follows:

(25)
$$\textit{quant-relation}: \begin{bmatrix} \text{ARG0} & \textit{ref-ind} \\ \text{RSTR} & \textit{handle} \\ \text{BODY} & \textit{handle} \end{bmatrix}$$

Since all quantifiers should need only these features, particular quantifiers should be distinguished by their PRED values (e.g., **_every_q_rel**, **_some_q_rel**, **_the_q_rel** in the ERG). The Matrix does not yet adopt any hypothesis about whether the inventory of quantifier relations might be made language-independent, though it would clearly be desirable if such an inventory could be defined.

4.2.2 Messages

We provide the *message* type in the Matrix as an MRS encoding of the distinctions among types of semantics of clauses developed in Ginzburg and Sag 2000. Clauses can express commands, questions, or propositions, either as simple sentences or in embedded contexts as sentential complements, relative clauses, etc. The type *message* is a subtype of *relation* which introduces the attribute MARG (message argument), whose value will be the highest scoping handle of the clause. The PRED value of the message relation is used to distinguish the three clause types, drawing from the pred sorts given in the hierarchy in Figure 5, all of which carry the suffix *_m_rel* for quick identification.

This type hierarchy has a bit more structure than might otherwise be expected as we anticipate that the *messsage* type of a clause (through CONT.MSG) will used in semantic selection by clause-taking verbs. As some verbs (like *know*) accept sentential complements

which are either propositions or questions but not commands, the Matrix provides the intermediate type *prop-or-ques_m_rel* which the verb *know* can use to constrain its complement's MSG.PRED value. Among questions, all of which will be assigned the MSG.PRED value *abstr-ques_m_rel* by the grammar, we distinguish between ordinary questions with MSG.PRED value *question_m_rel*) and the type of confirmation-seeking question expressed by a tag question in English, the particle *ne* in Japanese, the particle *ne* in German, or the equivalent in other languages, for which the Matrix supplies the pred sort *ne_m_rel*.[12]

The ERG has adopted the further assumption that all questions contain an immediately embedded proposition, motivated both by (reasonable but perhaps controversial) hypotheses about inference in discourse, and by concerns for simplicity in the hierarchy of constructions for English. This choice of implementation for the semantics of questions is supported by the Matrix, but not required by it.

4.2.3 Subordination and Coordination

The Matrix provides two relation types for subordination and coordination, both subtypes of *subord-or-conj-relation*, which introduces two new attributes for their semantic arguments. The first subtype *subord-relation* can be used for subordinating conjunctions like English *since*, and the second subtype *conjunction-relation* is used by ordinary conjunctions like English *and*. Both coordination and subordination relations take two handles as arguments; in the case of subordination, these correspond to the LTOP values of the main and subordinate clauses, while for coordination they correspond to the LTOP values of the two conjuncts. These are the values of the features L-HNDL and R-HNDL, with the names chosen to encode explicitly the order of left and right conjuncts, since this order can be semantically (or pragmatically) relevant at least for coordination.

(26) *subord-or-conj-relation*: $\begin{bmatrix} \text{L-HNDL} & \textit{handle} \\ \text{R-HNDL} & \textit{handle} \end{bmatrix}$

The main relation introduced by specific subordinating conjunctions (e.g., *if*, *because*, *while*) will be simply the general relation *subord-relation*, with each lexical entry providing distinguishing values for the PRED attribute. There is, however, more to be said about coordination. Coordination of clauses, predicates or noun phrases requires the construction of a single *individual* (*event* or *ref-ind*) that can be the value of a role feature in any outside predicate that takes the conjoined entity as an argument. This is achieved by giving *conjunction-relation* the three additional features shown in (27):

(27) $\begin{bmatrix} \text{C-ARG} & \textit{conj-index} \\ \text{L-INDEX} & \textit{index} \\ \text{R-INDEX} & \textit{index} \end{bmatrix}$

The value of C-ARG is the conjoined index (*conj-index*) which serves as a pointer to the separate conjoined entity.

[12] On *ne_rel*, see Bender and Flickinger 1999.

Of course, it is possible to conjoin more than two noun phrases, predicates or clauses. In order to keep a determinate number of features for any given relation, we represent multi-coordination via a series of chained binary *conjunction-relation*s, where the R-INDEX of all but the lowest *conjunction-relation* is the C-ARG of the next *conjunction-relation*.

Note that in the coordination of noun phrases, the L-HNDL and R-HNDL values of the *conjunction-relation* will be identified with the LTOP values of each of the two noun phrases, but these handle values will not be identified with the LBL values of any relations in the phrase, given our treatment of quantifiers as underspecified with respect to scope. In all other cases of coordination (clauses, VPs, predicative phrases, etc.), the values of L-HNDL and R-HNDL will be identified with the LBL values of relations in the phrase.

4.2.4 Miscellaneous Special Relations

In order to illustrate some ways in which this hierarchy of semantic relations might be extended, the Matrix provides three additional relations which grammar writers may find useful. Perhaps the most interesting of these is the *unspec-compound-relation*, intended for use in the construction of noun-noun compounds. These are treated in the ERG as a syntactic constituent with two noun daughters that are combined by a rule which introduces this *unspec-compound-relation*, a subtype of *arg12-relation*. This relation does not introduce any new attributes, but rather constrains the two arguments to both be of type *ref-ind*. Such a constraint might have value, but if not, the grammar writer could simply use the relation type *arg12-relation*, and assign the appropriate unique PRED value.

Similarly, the Matrix supplies the *verb-ellipsis-relation* as a subtype of *arg1-relation*, intended for analyses of elided verb phrases as in the English example *We tried to but we couldn't*. Here neither new attributes nor specific constraints on existing attributes are added, though we anticipate that the semantics of elliptical constructions in languages might require such additional machinery. Again, the grammar writer may well choose to ignore this type.

The third such type is the *prep-mod-relation*, a subtype of *arg12-ev-relation*, and intended for use with prepositions that head modifier phrases. Here again, no new constraints enrich this type, though for a given language it may be useful to impose additional type constraints on the values of ARG1 and ARG2 that would hold for all lexical entries and constructions introducing this relation type.

4.2.5 User-Defined Special Relations

The Matrix design anticipates that grammar writers may choose to extend the inventory of relation types to accommodate language-specific phenomena. For example, an analysis of degree specifiers and measure phrases like the English examples *the building was very tall* or *the building was fifty meters tall* may lead the grammar writer to a new relation type for these degree phrases. Similarly, comparative constructions, like in the English examples *Kim is taller than Sandy was* or *dogs have more legs than ostriches*, may require the introduction of relations with additional attributes to encode all of the relevant constraints.

When adding a new relation, be sure to place it correctly in the *relation* hierarchy so that it inherits exactly the attributes and constraints that it needs from existing types. And before choosing to make the addition, consider whether an existing relation type might in

fact serve, given the existence of the Sem-I component (described in §8 below) to provide a unique interpretation of the attributes ARG1 ... ARG4 for each relation.

5 Features of Indices: Agreement and TAM

The Matrix currently provides only minimal support for more fine-grained constraints on semantic indices, in part due to the high degree of variation needed across grammars to encode constraints on attributes like person, number, and gender for referential indices, or tense, aspect, and mood for events. Even a seemingly uncontroversial step like introducing separate attributes for each of these properties on indices has proven to be unproductive, since a given language may conflate two such attributes morphologically and syntactically.

For example, the ERG captures the distribution of person and number in English by introducing types which conflate these two properties, enabling a direct expression of the relevant syntactic contrasts without requiring a disjunctive representation of the constraints on subject-verb agreement. The ERG introduces an attribute of the type *png* called PN (person-number), whose values are of the type *pernum* defined as in Figure 6. Given these types, the constraint on the AGR value for non-third-singular verbs is simply

(28) $\begin{bmatrix} \textit{non3sg-verb} & \\ \text{AGR} & \begin{bmatrix} \text{PN} & \textit{non3sg} \end{bmatrix} \end{bmatrix}$

since the type *non3sg* will unify with any of its subtypes, including the AGR values *1sg, 2sg, 1pl, 2pl*, and *3pl*.

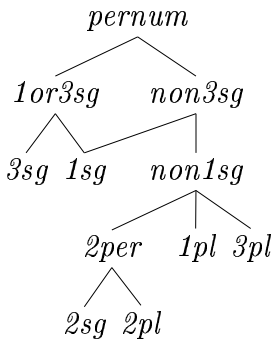


Figure 6: Hierarchy of PN values in the ERG

This generalization would be difficult to express without disjunctions if person and number were distinct attributes, and we expect similar conflations in particular languages for other agreement attributes, so we have left the internal structure of the values for the attributes PNG and TAM unspecified in the Matrix. In future work, we hope to include possible

tense/aspect systems as modules in the Matrix, i.e., as components that we expect to be useful in many but not all languages.

The treatment of tense, aspect, and mood is equally underspecified in the current version of the Matrix, though it is expected that a more elaborated candidate set of types will emerge in the near future, at least for some aspectual distinctions, growing out of work on the NorSource grammar of Norwegian.

6 Syntax-Semantics Interface

Section 3 discussed the basic semantic objects defined in the Matrix. Section 4 discussed types of relations and the representation of arguments of relations. The preceding section sketched our approach to representing agreement and TAM. This should give a fairly complete picture of the kinds of semantic representations Matrix-derived grammars should build. This section addresses how to go about building those representations compositionally as words are combined into successively larger phrases in the syntax. This section is organized as follows: §6.1 describes the implementation of general semantic principles. §6.2 describes the identification of semantic heads. §6.3 discusses constructions and lexical rules that make semantic contributions beyond simply combining the semantics of their daughters. §6.4 describes the linking of syntactic and semantic arguments within the lexicon, and the role of syntactic rules in associating the indices of syntactic arguments with the appropriate semantic requirements of a predicate. §6.5 describes how to ensure that all nominal indices are bound by an appropriate quantifier. §6.6 provides examples of handle constraints, including constraints on the RSTR values of quantifiers, how messages interact with handle constraints, and handle constraints concerning scopal modifiers like *probably*. Finally, §6.7 describes mismatches between syntax and semantics, including control constructions, expletives, and raising of KEY and INDEX values (e.g., by the English auxiliary *do*).

6.1 Semantic Principles

With every word or phrase providing a semantics which consists of HOOK, RELS, and HCONS, the principles of semantic composition in phrase structure rules can be stated (and implemented) quite elegantly, following the definitions in Copestake et al. 2001:

1. The value for RELS on the mother of a phrase is the result of appending the RELS values of all of its daughters.

2. The value for HCONS on the mother of a phrase is the result of appending the HCONS values of all of its daughters.

3. The value for HOOK on the mother of phrase is identified with the HOOK value of its semantic head daughter, where each phrase type uniquely determines which of the daughters is the semantic head.

In the Matrix, principles 1 and 2 are implemented as constraints on a few high-level types (*lex-rule*, *basic-unary-phrase* and *basic-binary-phrase*) within the *sign* subhierarchy (sketched

in Figure 7), such that they are inherited by all phrases and lexical rules. In addition, the type *phrase-or-lexrule* identifies the mother's HOOK with the HOOK of the semantics provided by the rule itself (the value of a feature called C-CONT; see §6.3 below). More specialized phrase types identify the HOOK of the C-CONT with the HOOK of the head or non-head daughter, as appropriate.

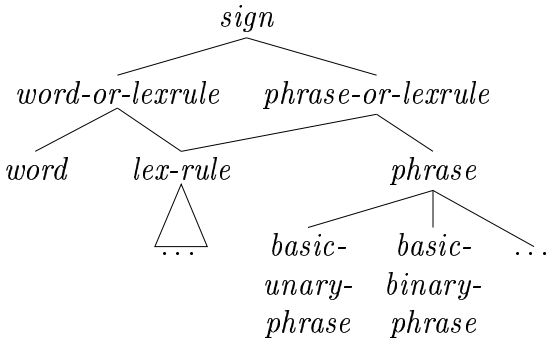


Figure 7: Subhierarchy under *sign*

Principles 1 and 2 require the monotonic accumulation of RELS and HCONS values from daughter to mother in a phrase. The values of these features are implemented as difference lists (typed feature structures which bear values for two attributes LIST and LAST), which allows us to state the accumulation of values using the same single operation of unification of typed feature structures. (29) shows the constraints on the type *basic-unary-phrase*, including the 'diff-list appends' that implement principles 1 and 2.

(29)

basic-unary-phrase:

$$\left[\begin{array}{l}
\text{SYNSEM.LOCAL.CONT}\left[\begin{array}{ll}\text{RELS} & \left[\begin{array}{ll}\text{LIST} & \boxed{1}\\ \text{LAST} & \boxed{2}\end{array}\right]\\ \text{HCONS} & \left[\begin{array}{ll}\text{LIST} & \boxed{4}\\ \text{LAST} & \boxed{5}\end{array}\right]\end{array}\right]\\
\text{C-CONT}\left[\begin{array}{ll}\text{RELS} & \left[\begin{array}{ll}\text{LIST} & \boxed{3}\\ \text{LAST} & \boxed{2}\end{array}\right]\\ \text{HCONS} & \left[\begin{array}{ll}\text{LIST} & \boxed{6}\\ \text{LAST} & \boxed{5}\end{array}\right]\end{array}\right]\\
\text{ARGS}\left\langle\left[\ldots\left[\text{CONT}\left[\begin{array}{ll}\text{RELS} & \left[\begin{array}{ll}\text{LIST} & \boxed{1}\\ \text{LAST} & \boxed{3}\end{array}\right]\\ \text{HCONS} & \left[\begin{array}{ll}\text{LIST} & \boxed{4}\\ \text{LAST} & \boxed{6}\end{array}\right]\end{array}\right]\right]\right]\right\rangle
\end{array}\right]$$

6.2 HOOK Features and Semantic Heads

The type *head-compositional* (a subtype of *headed-phrase*) provides the constraint that identifies the HOOK of the C-CONT (and therefore also of the mother) with the HOOK of the head-daughter:

(30) *head-compositional*: $\begin{bmatrix} \text{C-CONT.HOOK} & \boxed{1} \\ \text{HEAD-DTR} & \begin{bmatrix} \text{SYNSEM.LOCAL.CONT.HOOK} & \boxed{1} \end{bmatrix} \end{bmatrix}$

The types *basic-head-comp-phrase*, *basic-extracted-comp-phrase*, and *basic-head-opt-comp-phrase* (and therefore its two subtypes) all inherit this constraint from *head-compositional*. That is, these types of phrases are all cases where the syntactic head and the semantic head are the same. Note that *basic-extracted-comp-phrase* and *basic-head-opt-comp-phrase* are unary phrases, but it is still necessary to determine whether or not they are head-compositional, as there are also unary phrases which contribute constructional semantics and therefore do not identify the HOOK of the C-CONT with the HOOK of the daughter (see §6.3 below).

The type *head-mod-phrase* does not inherit from *head-compositional*, but one of its subtypes *isect-mod-phrase* does, to account for the semantic properties of phrases with intersective modifiers. In contrast, one of its other subtypes *scopal-mod-phrase* imposes the opposite constraint, identifying the HOOK values of the C-CONT and the non-head daughter, as required for the semantics of phrases containing scopal modifiers like *probably*. Thus we treat the modifier in scopal head-modifier constructions as the semantic head, even though it is not the syntactic head. Similarly, in *basic-head-spec-phrase* the HOOK of the C-CONT is identified with the HOOK of the specifier daughter, as we take specifiers to be semantic heads.

The third possibility is illustrated in the type *head-subj-phrase*, which supplies a message relation that takes widest scope, and hence the C-CONT.HOOK of the phrase is not identified with that of either daughter. Instead, the C-CONT.HOOK.LTOP value is identified with the handle (label) of the newly supplied message relation, and the C-CONT.HOOK.INDEX value is identified with that of the head daughter. See the next section for a discussion of how constructions can contribute semantics.

So we have (syntactically) headed phrases where the semantic head is the syntactic head, headed phrases where the semantic head is instead the (syntactic) non-head daughter, and headed phrases where the C-CONT is the semantic head. The Matrix does not provide any subtypes of *non-headed-phrase*, but every phrase must have some semantic head, even if only the C-CONT. The constraints on *lex-rule*, *basic-binary-phrase*, and *basic-unary-phrase* ensure that the HOOK of the mother will always be identified with the HOOK of the C-CONT, but it is the responsibility of the grammar developer to make sure that the HOOK of the C-CONT provides sufficient information. This will be ensured if it is identified with the HOOK of a daughter, or if the C-CONT has a non-empty RELS value and the features inside HOOK are related to the appropriate parts of some *relation* on RELS.

The next subsection provides some examples of semantically contentful constructions.

6.3 Semantic Contributions of Constructions

Since some phrase types may introduce semantic content which is not drawn from any of the daughters of the phrase, the MRS framework provides an attribute for phrasal signs called C-CONT (for construction content), which behaves with respect to the semantics principles just as though it were another daughter of the phrase (see, for example, (29) above). C-CONT is implemented in the Matrix as a top-level attribute of phrases and lexical rules, introduced on the type *phrase-or-lexrule*. Like CONT, its value is of type *mrs*.

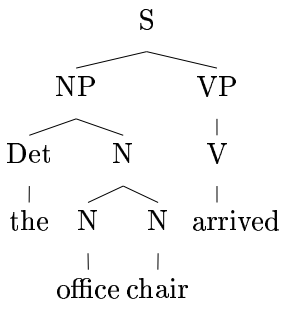
If a phrase does not introduce any additional semantic content of its own, the values for the attributes RELS and HCONS in C-CONT will be empty lists, so unary and binary phrases can safely always append these values to those supplied by the syntactic daughters. Likewise, the HOOK value of a phrase is always identified with its C-CONT's value for HOOK, where for most phrases this HOOK in C-CONT will simply be identified with that of one of the daughters of the phrase, namely the semantic head daughter.

6.3.1 Syntactic Constructions

One example of a phrase type in the Matrix that does introduce its own semantic content is the type for (non-relative) clauses (*non-rel-clause*), which introduces a relation encoding the illocutionary force of the clause. Such relations (e.g., *prpstn_m_rel*, mentioned above) are of type *message*, following the analysis in (Ginzburg and Sag 2000).

We illustrate construction-introduced semantic content further with the treatment of noun-noun compounds in the ERG. In the analysis of the sentence *the office chair arrived*, the phrase *office chair* is built using a syntactic rule specifically for noun-noun compounds, and this rule introduces a generic two-place relation **n-n-cmpnd_rel** which relates the variables introduced by the two nouns. The syntactic structure is sketched in (31), where the head-specifier rule is used to combine the determiner and the compound noun, while the head-subject rule combines the full NP with the verb phrase *arrived*. The corresponding MRS semantics is shown in (32):

(31)



The office chair arrived

(32)
$$\left[\begin{array}{l} \textit{mrs} \\ \textsc{hook} \quad \left[\begin{array}{ll} \textsc{ltop} & \textit{h1} \\ \textsc{index} & \textit{e2} \end{array}\right] \\ \textsc{rels} \quad \left\langle ! \left[\begin{array}{ll} \textbf{proposition_m_rel} & \\ \textsc{lbl} & \textit{h1} \\ \textsc{marg} & \textit{h4} \end{array}\right], \left[\begin{array}{ll} \textbf{_the_q_rel} & \\ \textsc{lbl} & \textit{h10} \\ \textsc{arg0} & \textit{x8} \\ \textsc{rstr} & \textit{h11} \\ \textsc{body} & \textit{h12} \end{array}\right], \left[\begin{array}{ll} \textbf{_chair_n_rel} & \\ \textsc{lbl} & \textit{h7} \\ \textsc{arg0} & \textit{x8} \end{array}\right], \left[\begin{array}{ll} \textbf{_office_n_rel} & \\ \textsc{lbl} & \textit{h14} \\ \textsc{arg0} & \textit{x16} \end{array}\right], \right. \\ \qquad \left[\begin{array}{ll} \textbf{udef_q_rel} & \\ \textsc{lbl} & \textit{h17} \\ \textsc{arg0} & \textit{x16} \\ \textsc{rstr} & \textit{h18} \\ \textsc{body} & \textit{h19} \end{array}\right], \left[\begin{array}{ll} \textbf{n-n-cmpnd_rel} & \\ \textsc{lbl} & \textit{h7} \\ \textsc{arg1} & \textit{x8} \\ \textsc{arg2} & \textit{x16} \end{array}\right], \left[\begin{array}{ll} \textbf{_arrive_v_rel} & \\ \textsc{lbl} & \textit{h21} \\ \textsc{arg0} & \textit{e2} \\ \textsc{arg1} & \textit{x8} \end{array}\right] \left. ! \right\rangle \\ \textsc{hcons} \quad \left\langle ! \left[\begin{array}{ll} \textit{qeq} & \\ \textsc{harg} & \textit{h4} \\ \textsc{larg} & \textit{h21} \end{array}\right], \left[\begin{array}{ll} \textit{qeq} & \\ \textsc{harg} & \textit{h11} \\ \textsc{larg} & \textit{h7} \end{array}\right], \left[\begin{array}{ll} \textit{qeq} & \\ \textsc{harg} & \textit{h18} \\ \textsc{larg} & \textit{h14} \end{array}\right] ! \right\rangle \end{array}\right]$$

This MRS representation contains two noun relations each associated with a quantifier relation to bind the variables that are their ARG0 values. The two noun variables are also identified with the ARG1 and ARG2 attributes of the **n-n-cmpnd_rel** relation, and the variable for the head noun *chair* is also the value of the single argument of the *arrive* relation. The *n-n-cmpnd* relation is introduced by the grammar in the RELS attribute of the C-CONT of the grammar rule for noun-noun compounds, which also identifies the assignments of the two nominal instance variables (supplied by its two daughters) to the ARG0 and ARG1 attributes of that *n-n-cmpnd* relation. The relevant constraint on the grammar rule is sketched in (33):

(33)
$$\left[\begin{array}{l} \textsc{head-dtr...hook} \quad \left[\textsc{index} \ \boxed{1}\right] \\ \textsc{non-head-dtr...hook} \quad \left[\textsc{index} \ \boxed{2}\right] \\ \textsc{c-cont} \quad \left[\textsc{rels.list} \left\langle \left[\begin{array}{ll} \textit{n-n-cmpnd} & \\ \textsc{arg1} & \boxed{1} \\ \textsc{arg2} & \boxed{2} \end{array}\right] \right\rangle\right] \end{array}\right]$$

As discussed above, general principles of semantic composition that are encoded in the Matrix types ensure that rule-specific relations are gathered up along with the relations supplied by the daughters of the rule, and that the appropriate external semantic hooks (the LTOP and INDEX values) are identified on the phrase itself, ready for further composition.

6.3.2 Lexical Rules

Lexical rules are treated in the Matrix as a particular type of unary rule, in most respects like syntactic unary rules, though lexical rules are prevented from interleaving with syntactic

rules. Thus semantic composition for lexical rules is implemented using the same principles as outlined above for syntactic phrases. A lexical rule may or may not introduce semantic content of its own; if it does, that content is found in the C-CONT attribute of the rule and is combined with the semantic content of the (single) daughter (the 'input' to the lexical rule) by those same principles.

Note that this approach imposes a strong constraint on the directionality of lexical rules in the Matrix, since the principles of composition guarantee monotonic accumulation of atomic predications, so no semantic content from a daughter in a phrase or lexical rule is ever lost. For example, a lexical rule relating the causative/inchoative alternation in English for verbs like *break* will have to treat the inchoative lexical entry as the 'input' to the lexical rule (that is, the daughter), and the semantically richer causative lexical entry as the 'output' (the SYNSEM value of the lexical rule).

6.4 Linking

Lexical entries can select syntactic arguments, such as complements, subjects, and specifiers, constraining the properties of these arguments to capture the relevant subcategorization requirements. At the same time, these lexical entries can impose constraints which determine the way that the semantics of their arguments will combine with the semantics of the selecting entry. These constraints linking the semantic hooks of syntactic arguments to the appropriate semantic argument positions are introduced in lexical entries and interact with corresponding constraints in the constructions that combine a word or phrase with one or more of its syntactic arguments.

For example, a transitive verb like English *chase* subcategorizes for an NP subject and an NP object, where in the ERG the verb combines with its object using the *head-complement* rule, and with its subject using the *head-subject* rule. The semantic index of the subject NP is assigned to the ARG1 role in the **_chase_v_rel** relation, while the index of the object NP is assigned to the ARG2 role, where the interpretation of these role names (as the chaser and the thing chased, respectively) is provided by a separate component of the grammar called the Sem-I, described in §8.

The linking type for transitive verbs like *chase* in the ERG includes the following simple linking constraints:[13]

(34)
$$\begin{bmatrix} \textit{trans-lt} & \\ \text{SUBJ} & \langle \left[\text{HOOK.INDEX} \quad \boxed{1}\right] \rangle \\ \text{COMPS} & \langle \left[\text{HOOK.INDEX} \quad \boxed{2}\right] \rangle \\ \text{KEYREL} & \begin{bmatrix} \text{ARG1} & \boxed{1} \\ \text{ARG2} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

[13] These constraints are not stated directly on the type *trans-lt*, but are rather inherited from its supertypes. We display them on *trans-lt* for expository convenience. In future versions of the Matrix, we expect to state such constraints using the feature ARG-S (Argument Structure) rather than valence features like SUBJ or COMPS.

When the verb phrase *chased the cat* is constructed using the *head-complement* rule, the feature structure for the sign *the cat* is unified with the constraints in the COMPS attribute of *chase*, including both the syntactic requirements for an accusative NP and the semantic constraints which identify the semantic index of that NP with the ARG2 role in the head's relation **_chase_v_rel**. An analogous identification is made when the subject NP *the dog* is combined with the verb phrase *chased the cat* using the *subject-head* rule, so that the NP's semantic index is unified with the ARG1 role of the **_chase_v_rel**.

Lexical entries like the English verb *insist* which take sentential complements differ from simple transitive verbs in that they impose a linking constraint on the local top LTOP attribute of their complement rather than on the INDEX of that complement. This ensures that the semantics of the embedded sentence falls within the scope of the semantic relation introduced by the selecting verb, as discussed in §6.6.1 below.

The lexical type for verbs like *insist* as in *Kim insisted that Sandy was right* in the ERG includes the following linking constraints:[14]

(35)
$$\begin{bmatrix} \textit{cp-intrans-verb} & \\ \text{SUBJ} & \langle \left[\text{HOOK.INDEX} \ \boxed{1}\right] \rangle \\ \text{COMPS} & \langle \left[\text{HOOK.LTOP} \ \boxed{2}\right] \rangle \\ \text{KEYREL} & \begin{bmatrix} \text{ARG1} & \boxed{1} \\ \text{ARG2} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

Lexical entries for modifiers introduce similar linking constraints on the words or phrases that they modify, even though modifiers are not the syntactic heads in head-modifier constructions. These constraints on both the syntax and semantics of modified phrases are introduced in the HEAD.MOD attribute of a modifier's lexical entry (instead of the valence features including SUBJ or COMPS), and the constructions that combine modifiers and heads unify the feature structure of the head with the constraints in the HEAD.MOD attribute of the modifier, analogous to the effects of the valence-combining constructions.

For example, the lexical type for simple intersective adjectives like English *tall* includes the following linking constraints:

(36)
$$\begin{bmatrix} \textit{adj-synsem} & \\ \text{HEAD.MOD} & \langle \left[\text{HOOK.INDEX} \ \boxed{1}\right] \rangle \\ \text{KEYREL} & \left[\text{ARG1} \ \boxed{1}\right] \end{bmatrix}$$

Then the *intersective-modifier-head* rule which combines *tall* with *chair* unifies the HEAD.MOD constraints of *tall* with the feature structure for *chair*, and in addition the rule identifies the LTOP values of the two daughters, ensuring that the noun and its modifying adjective are assigned the same scope since they have a common handle.

Scopal modifiers like the English adverb *probably* impose a linking constraint on the phrase they modify which is analogous to that introduced by verbs like *insist*, making reference to the modified phrase's LTOP attribute rather than its INDEX:

[14] See note 13.

(37) $\begin{bmatrix} \textit{basic_scopal_adverb_synsem} & \\ \text{HEAD.MOD} & \langle \left[\text{HOOK.LTOP} \quad \boxed{1}\right] \rangle \\ \text{KEYREL} & \left[\text{ARG1} \quad \boxed{1}\right] \end{bmatrix}$

The construction that combines scopal modifiers with their heads will unify this constraint from the modifier on the head's LTOP, resulting in the semantics of the modified phrase falling within the scope of the relation introduced by the modifier, as desired.

6.5 Indices Bound by Quantifiers

One of the requirements for an *mrs* to be well-formed is that each of the referential indices (*ref-ind*) must be bound by a quantifier. In the Matrix this means that within a sentence each time a noun phrase is constructed which introduces a referential index (the ARG0 value of the relation supplied by the head noun), that ARG0 value must be identified with the ARG0 value of exactly one *quant-relation* in the *mrs* for the sentence. In addition, if the value of an argument role in some relation ARG1 ... ARG4 is constrained to be of type *ref-ind*, then that variable will have to be bound by a quantifier. This means that if the grammar writer defines a lexical entry for a verb with optional arguments, the values of the ARGN roles in that verb's relation should be left underspecified when defining the lexical entry for the verb. For example, the English verb *eat* might be defined as a transitive verb whose NP object is optional, allowing both *The mouse ate* and *The mouse ate the cheese.* The value of the ARG2 role in the **arg12-ev-relation** for this optional direct object can be constrained to be of type *index*, but not *ref-ind*, in the definition of the lexical entry for *eat.* If the direct object is not present, the ARG2 value will then be treated as unbound, and if the direct object is present, it will supply both the referential index and the quantifier that binds it.

Note that semantic variables of type *event* in the Matrix are not bound by quantifiers, so they do not affect the well-formedness of an *mrs*, even if they appear as the argument of some predicate. And indices introduced by expletive pronouns, which will be (subtypes of) *expl-ind*, are not expected to appear as arguments of relations in an *mrs.*

6.6 Imposing Handle Constraints

Since the elementary predications for a phrase or sentence are simply collected up as a bag or unordered list in the attribute CONT.RELS, any scope relations among these predications must be expressed by means of constraints on their handles, the value of the attribute LBL in each elementary predication. In this subsection we discuss the three most typical kinds of handle constraints used in Matrix grammars.

6.6.1 Embedded Clauses

A clause is taken in the Matrix to be a saturated projection of a verb (or perhaps some other tense-bearing predicate) which introduces a message relation (a proposition, question, or command). As discussed above, each message relation introduces just one argument role

called MARG, whose value will constrain the handle with the next-highest scope in that clause (ignoring quantifiers), by means of a *qeq* constraint (cf. §3.3). This next-highest scoping handle will typically be the value of the LBL attribute of the head verb's relation. For simplex sentences with only one clause, the LBL value of the one message relation will be identified with the LTOP of the semantics for the whole sentence, expressing the fact that this message has scope over every other relation within the sentence, including all quantifiers.

When one clause is embedded as an argument or modifier phrase within a larger sentence, the handle of its message relation is instead identified either with an argument role of the embedding predicate, or with the LTOP of the phrase it is modifying. This expresses the fact that the semantic content of this embedded clause falls within the scope of some other relation in the sentence. We can illustrate with an example of each of these types of embedded, first for sentential complements and then for modifier phrases.

In the English sentence *Kim wondered whether Sandy arrived*, the clause *whether Sandy arrived* is a complement selected by the verb *wonder*, which introduces an **arg12-ev-relation** whose ARG1 value is the referential index supplied by *Kim* and whose ARG2 value is the LTOP of the embedded clause. This LTOP for *whether Sandy arrived* is identified with the LBL of the message relation (of type *question*) as a result of the semantic composition of that clause.

Relative clauses are taken to be intersective modifiers of nouns, and as we have already seen, intersective modification is expressed in the Matrix by identifying the LTOP values of the modifier and the phrase being modified. Thus in the sentence *Every dog that the cat chased barks*, the LTOP of the relative clause *that the cat chased* is identified with the LTOP of the noun *dog*, expressing the fact that the semantic content of the relative clause will fall inside the scope of the quantifier *every* binding the referential index supplied by *dog*.

6.6.2 Quantifiers and Scope

As seen above, each referential index introduced in an *mrs* must be bound by a quantifier, but in addition, the handle of the noun relation introducing that *ref-ind* must also be appropriately constrained. Each nominal phrase that is ready to combine with a determiner will identify its LTOP with the handle of the highest-scoping elementary predication in that phrase, typically the LBL of the head noun. When this phrase becomes a noun phrase, either by combining with a determiner or via some construction which supplies the quantifier, this LTOP value of the nominal phrase might be expected to be identified with the RSTR (restrictor) value of its quantifier. But in order to allow for the full range of possible scope relations within noun phrases, a *qeq* handle constraint is introduced, where the RSTR value of the quantifier relation is identified with the HARG (the "higher" scope position) of the *qeq*, and the LTOP of the N-bar is identified with the LARG (the "lower" scope position). This seemingly cumbersome introduction of a *qeq* for the semantics within every noun phrase ensures that all possible readings of the phrase are correctly represented, allowing just the right range of variation in quantifier scope. For a fuller discussion of these issues, see Copestake et al. 2003.

6.6.3 Scopal Modifiers

Some modifiers like the English adverb PROBABLY are treated as scopal, which means that instead of taking as their argument the *ref-ind* or *event* of the phrase they modify, they identify their semantic relation's argument position with the top handle of the phrase they modify. This expresses the fact that the semantic content of the modified phrase falls within the scope of the semantic predicate introduced by the modifier. As with the other examples of scopal interactions we have seen, here too we want to allow for intervening quantifiers in the underspecified representation for a sentence like *Every manager will probably hire some consultants*. So the lexical entry for *probably* does not identify the LTOP of the verb phrase it modifies directly with its ARG0, but rather introduces a *qeq* relation whose HARG is identified with the adverb's ARG0 and whose LARG is identified with the LTOP of its MOD value.

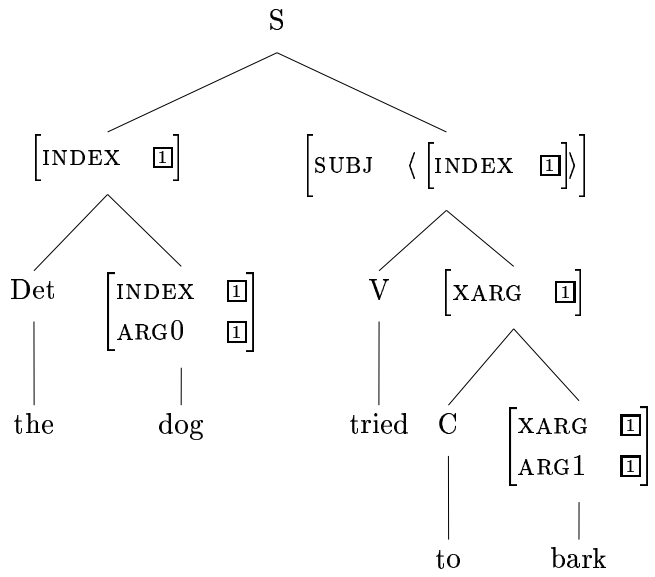
6.7 Syntax-Semantics Mismatches

When defining the linking between syntactic arguments of a lexical entry and the corresponding semantic argument positions in the relation introduced by that entry, a grammar writer will often encounter mismatches. A single syntactic argument may have its index be linked to a position in more than one head's semantics (as in equi constructions), or it may not appear in the head's semantic relation at all, as in raising or expletive constructions. In this section we describe the mechanisms provided in the Matrix for defining these mismatches in the syntax-semantics interface.

6.7.1 External arguments and control

We have made reference so far to two of the *hook* attributes, LTOP and INDEX, both of which play a crucial role in the semantic construction of every phrase. A third attribute, XARG, is relevant for control phenomena such as equi and raising, since it identifies the semantic index of a phrase's external argument (usually the subject of a verb phrase). Identifying this property of a phrase as part of the hook allows our general principles of semantic composition to make this attribute visible for control of subject-unsaturated complements (VPs, predicative PPs, etc.) and also for agreement even at the sentence level, as for example in tag questions in English (Bender and Flickinger 1999). An example using this XARG attribute in composition is given in (38), with the lexical type for subject-equi verbs given in (39).

(38) The dog tried to bark.



(39) Type for subject-equi verbs like *try*

$$\begin{bmatrix} \textit{subj-equi-verb} & \\ \textsc{subj} & \langle \boxed{1} \rangle \\ \textsc{comps} & \langle \boxed{2} \rangle \\ \textsc{keyrel} & \boxed{3} \end{bmatrix}$$

Here the lexical entry for the verb *try* identifies its VP complement's semantic external argument (XARG value) with its subject's semantic index (INDEX value), and further identifies that index with the appropriate role (the ARG1) in the lexical relation introduced by the verb. The MRS semantics constructed for this example is given in (40).

(40)
$$\begin{bmatrix} \textit{mrs} \\ \textsc{hook} & \begin{bmatrix} \textsc{ltop} & \textit{h1} \\ \textsc{index} & \textit{e2} \end{bmatrix} \\ \textsc{rels} & \langle ! \begin{bmatrix} \textbf{prpstn_m_rel} \\ \textsc{lbl} & \textit{h1} \\ \textsc{marg} & \textit{h4} \end{bmatrix}, \begin{bmatrix} \textbf{def_q_rel} \\ \textsc{lbl} & \textit{h10} \\ \textsc{arg0} & \textit{x8} \\ \textsc{rstr} & \textit{h11} \\ \textsc{body} & \textit{h12} \end{bmatrix}, \begin{bmatrix} \textbf{_dog_n_rel} \\ \textsc{lbl} & \textit{h7} \\ \textsc{arg0} & \textit{x8} \end{bmatrix}, \\ & \begin{bmatrix} \textbf{_try_v_rel} \\ \textsc{lbl} & \textit{h21} \\ \textsc{arg0} & \textit{e2} \\ \textsc{arg1} & \textit{x8} \\ \textsc{arg2} & \textit{h22} \end{bmatrix}, \begin{bmatrix} \textbf{prpstn_m_rel} \\ \textsc{lbl} & \textit{h22} \\ \textsc{marg} & \textit{h23} \end{bmatrix}, \begin{bmatrix} \textbf{_bark_v_rel} \\ \textsc{lbl} & \textit{h24} \\ \textsc{arg0} & \textit{e3} \\ \textsc{arg1} & \textit{x8} \end{bmatrix} ! \rangle \\ \textsc{hcons} & \langle ! \begin{bmatrix} \textit{qeq} \\ \textsc{harg} & \textit{h4} \\ \textsc{larg} & \textit{h21} \end{bmatrix}, \begin{bmatrix} \textit{qeq} \\ \textsc{harg} & \textit{h11} \\ \textsc{larg} & \textit{h7} \end{bmatrix}, \begin{bmatrix} \textit{qeq} \\ \textsc{harg} & \textit{h23} \\ \textsc{larg} & \textit{h24} \end{bmatrix} ! \rangle \end{bmatrix}$$

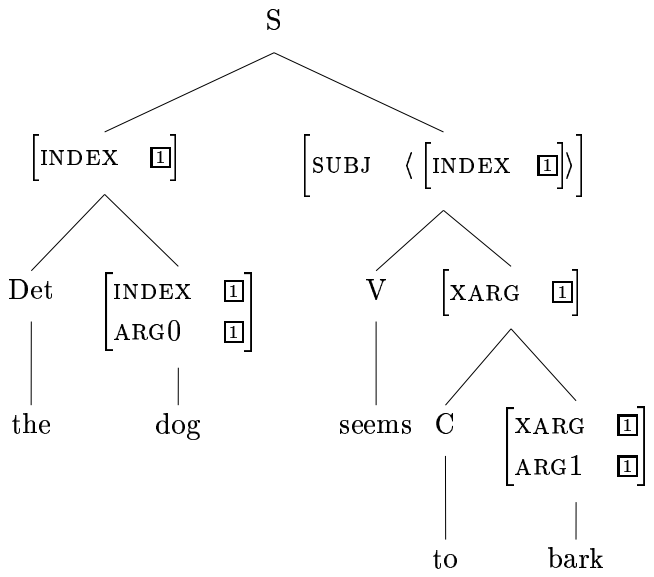
The construction of this representation is the result of the same general principles of semantic composition presented above. The *head-complement* rule unifies the verb *tried*'s constraints on its complement with those of the verb phrase *to bark*, which results in the identification of the XARG value of that VP with the INDEX of the subject of *tried*. The constraints on *tried*'s subject are propagated up to the verb phrase *tried to bark* from the head-daughter *tried* by the head-complement rule, and the semantics of this verb phrase preserve the semantic properties of its daughters, including the desired re-entrancies with the subject index. Hence when the head-subject rule combines *the dog* with *tried to bark*, the syntactic and semantic constraints of the noun phrase are unified with those in the SUBJ attribute of the verb phrase, resulting in the identification of the ARG0 value introduced by *dog* with the ARG1 values in both **_try_v_rel** and **_bark_v_rel**.

6.7.2 Raising

The same XARG attribute enables a straightforward representation of raising phenomena, where a syntactic argument's semantic index is not linked to any semantic argument position in a given lexical entry's semantic relation, but is instead assigned to a role in the semantics of another syntactic argument of the lexical entry. For example, the English verb *seem* can have two syntactic arguments, an NP and a VP, as in *the dog seems to bark*, but the semantics of *seem* introduces a one-argument relation **_seem_v_rel** which takes a proposition.

Compare the following parse tree and lexical type definition to the ones for the subject-control example with *try* above:

(41) The dog seems to bark.



(42) Type for subject-raising verbs like *seem*

$$\begin{bmatrix} \textit{subj-rais-verb} & \\ \textsc{subj} & \langle \left[\textsc{hook.index} \quad \boxed{1}\right] \rangle \\ \textsc{comps} & \langle \left[\textsc{hook} \begin{bmatrix} \textsc{ltop} & \boxed{2} \\ \textsc{xarg} & \boxed{1} \end{bmatrix}\right] \rangle \\ \textsc{keyrel} & \left[\textsc{arg1} \quad \boxed{2}\right] \end{bmatrix}$$

The difference between the lexical entries for *seem* and *try* is that the raising verb identifies the external argument of its VP complement with its own subject's index, but then takes the proposition introduced by the VP complement as its only argument, rather than assigning a second semantic role to its own subject's index.

Object-raising lexical entries like the English *believe* instantiate a lexical type similar to the one for subject-raising verbs, but identify the external argument of their VP complement with the index of their direct object NP rather than that of their subject. The relevant type is defined as follows:

(43) Type for object-raising verbs like *believe*

$$\begin{bmatrix} \textit{obj-rais-verb} & \\ \textsc{subj} & \langle \left[\textsc{hook.index} \quad \boxed{1}\right] \rangle \\ \textsc{comps} & \langle \left[\textsc{hook.index} \quad \boxed{2}\right], \left[\textsc{hook} \begin{bmatrix} \textsc{ltop} & \boxed{3} \\ \textsc{xarg} & \boxed{2} \end{bmatrix}\right] \rangle \\ \textsc{keyrel} & \begin{bmatrix} \textsc{arg1} & \boxed{1} \\ \textsc{arg2} & \boxed{3} \end{bmatrix} \end{bmatrix}$$

6.7.3 Expletives

Constructions with expletive arguments are analyzed similarly to raising constructions in that a lexical entry's syntactic argument introduces a semantic index which does not appear as a role in the entry's semantic predicate; indeed, that index appears in no relation in the semantics. For example, the English sentence *it rained* is analyzed in the ERG to have the semantics **_rain_v_rel**(e), with the lexical entry for *rain* requiring its subject NP to be the expletive *it*. The lexical type for verbs like *rain* introduces the following constraints, making use of an English-specific subtype of the Matrix *expl-ind* named *it-ind* which is introduced by the relevant lexical entry for the NP *it*.

(44)
$$\begin{bmatrix} \textit{expl-subj-verb} & \\ \textsc{subj} & \langle \left[\textsc{hook.index} \quad \textit{it-ind}\right] \\ \textsc{keyrel} & \left[\textsc{arg0} \quad \textit{event}\right] \end{bmatrix}$$

The lexical entry for the English presentational *be* as in the example *there is a dog here* requires its subject NP to have a semantic index of a second type of expletive, called *there-ind*, which is introduced by the relevant lexical entry for the NP *there*. The following example illustrates the interaction of expletives with object-raising predicates:

(45) The dog believes there to be a cat

7 Verifying *mrs*s

The LKB provides several useful tools for ensuring that the *mrs* representations that are built with a Matrix-derived grammar are well formed. These include batch tests for correctness of lexical entries and for the fully scoped readings of sentences, as well as alternate views of the *mrs*s for sentences. The basic functionality is documented in (Copestake 2002), so here we only highlight a few of the most common tests that a grammar writer can use.

First, it is important that the lexical entries for the grammar are all correctly defined, and one good batch test for this level of well-formedness can be invoked from the "LKB Top" window, by selecting the "Debug – Check lexicon" menu item. This will cause each lexical entry in the lexicon to be expanded and checked, with helpful error messages printed if any flaws are uncovered.

Second, the wellformedness of an *mrs* for a sentence can be checked by making sure that the underspecified semantics constructed by the grammar can be used to produce the right range of fully-scoped representations. This check can be invoked by left-clicking on one of the small parse trees that appears after successfully parsing a sentence, and selecting the menu item "Scoped MRS", which will either bring up a window showing all of the possible fully-scoped realizations for the chosen sentence, or produce an error message, often with helpful advice about what is wrong with the *mrs*.

A third useful view of an *mrs* can be invoked by again left-clicking on one of the small parse trees, and selecting the menu item "Dependencies", which will bring up a window showing a simplified view of the elementary dependencies that can be computed from the

input *mrs*. If the color of the text is blue, the argument links among predicates are all well-formed, whereas if the text color in the window is yellow, at least one link is not well-formed, and the line for the offending predicate is marked with an initial vertical bar to help in diagnosis. If the feature structure for the input *mrs* has defined a circular linking structure (usually because of an error in how the difference lists have been appended), then the text color in this "Dependencies" window will be red.

8 Sem-I: The Semantic Interface

In order to make use of a semantic representation produced by a Matrix grammar, an application developer will require a specification of how to interpret the MRS structures, including documentation of the full range of semantic relations and their arguments that the grammar introduces. This specification, called the Semantic Interface (Sem-I) should be supplied as part of each grammar, and will be the locus of some linguistic generalizations that one might have expected to see encoded directly in the lexical type hierarchy or in lexical rules.

The Sem-I will consist of:

1. A manually-constructed fully-documented database of all the relations introduced via constructions and lexical types, and all the values which may appear on semantic features. This is the meta-level Sem-I, and should be consistent across languages: i.e., it forms part of the Matrix.

2. An automatically constructed database for the semantic information pertaining to the open class words in each language. This is the object-level Sem-I. It is generated from the lexical database for each grammar's lexicon.

The Matrix will have associated code for generating the object-level Sem-I, for validating the well-formedness of *mrs*s constructed by the grammar, and for employing Sem-I mappings in displaying MRS representations.

8.1 The meta-level Sem-I

This is a manually constructed, fully documented database of all the relations introduced via constructions and lexical types, and all the values which may appear on semantic features. Since this inventory is intended to be consistent across languages, it forms part of the Matrix, with links to language-specific examples that illustrate the use of these relations and values for a particular grammar.

For example, a two-place relation introduced for noun-noun compounds in an English grammar derived from the Matrix might appear in the meta-level Sem-I database as follows:

PRED	ARG0		ARG1		ARG2		documentation	test suite
n-n-cmpnd_rel	event	obl	index	obl	index	obl	`<link to doc>`	`<ex. #>`

Here, the specification and instantiation of the types of the ARG0, ARG1, ARG2 values may be done automatically, as will be the case with the object-level Sem-I. Each semantic role

is identified as an obligatory ('obl') or optional ('opt') argument of the predicate, with the expectation that more fine-grained distinctions may be needed later. The documentation needs to explain the meaning and use of the relation in as much detail as necessary for application developers to decide how to treat it. For instance, it would be important for a developer to know whether **n-n-cmpnd_rel** was used for all noun-noun compounds in this grammar or only for some of them.

The meta-level Sem-I will also include specifications of semantic classes to provide the basis for generalizations over thematic roles, and to enable more mnemonic displays of MRS structures where the role names can be specialized for these semantic classes.

8.2 The object-level Sem-I

Each grammar built using the Matrix should eventually include an automatically constructed database for the semantic information pertaining to the open class words in the language. This is the object-level Sem-I, generated from the lexical database for each lexicon, with possible links to example sentences that can be used for testing.

For example:

lexeme	string	PRED	ARG0		ARG1		ARG2		test suite eg	doc
chase_v1	chase	_chase_v_rel	event	+	index	+	index	+	Dogs chased cats	"Doc"

In most cases there won't be a test suite example, but it may be that a developer will add one to elucidate the use or distinction from another sense. Similarly, documentation may be added, but usually won't be. The mechanism for adding documentation or test suite examples must make it possible to autogenerate the links in this database.

8.3 Thematic role mapping

The Matrix uses a naming convention where role names of the form ARGn are used to identify individual semantic arguments for predicates. While this has proven to be important in capturing generalizations about linking in the lexical type hierarchy, it requires that other linguistic generalizations over thematic roles be expressed instead in the Sem-I.

The approach planned for the Matrix is two-fold: a rich inventory of word classes will be incrementally developed which will allow mapping of ARGn relations into alternative inventories, and since this is a long term project, it might be augmented with automatically generated example sentences as a form of cheap documentation to convey the intent of role assignments for any given predicate.

8.4 Word classes

Information about word classes should be added incrementally to the lexical databases, without changing the grammars. Documentation for these classes explains the notion of ARGn with respect to that class, and mappings to alternative thematic roles can be done on the basis of class membership. Verb classes will be part of the Matrix Sem-I, since they are motivated by properties of the syntax-semantics interface. In contrast, thematic role

mappings are additions to the Sem-I which may be provided by developers of a particular grammar. The LKB software will support parameterized I/O routines which will allow alternative thematic role inventories to be supported if there is an available mapping.

For example, we can distinguish two classes of English psych predicates: one in which the subject is the experiencer of some emotion and the object the stimulus (e.g., *Kim likes rabbits*) and the other in which the converse is true (e.g. *Rabbits worry Kim*). The lexical database entries for *like* and *worry* can be enhanced to include their class.

A grammar developer wishing to add a semantic class to the Sem-I should provide the following information:

1. Class name.
2. Class documentation, to include class membership criteria in the form of specific tests and some specific exemplars.
3. Documentation of the ARGn behaviour of the class in the grammar.
4. A full list of all current lexical entries that are members of the class.
5. A mapping between the ARGn for this class and any supported thematic role sets.

8.5 Example sentences

Automatically created examples might be used to make the behaviour of lexical entries clearer to outside users. The idea is to augment entries which have subcategorization with some very coarse-grained selectional restriction information. For instance:

- handle
- event
- non referential
- referential
    1. animate
    2. non-animate physical
    3. physical location
    4. temporal location
    5. abstract

This would allow us to automatically construct standardized examples, which can be used to illustrate ARGs etc. For instance:

the person liked the thing
the person liked doing the thing
the person liked to do the thing

where *person* is the standard term for animate entities and *thing* for non-animate ones.

The point about this approach is to make it reasonably intuitive for application developers while avoiding making the process of deciding on the semantic categories too time-consuming. It will also simplify the task of checking lexical entries for over-generation, by allowing the grammar writer to scan the automatically generated examples for obviously incorrect sentences.

9 Conclusion

We have provided documentation here for the semantic types and their attributes that are used in the Matrix, as well as a discussion of how these types are used in semantic composition to produce well-formed meaning representations. We illustrated the concepts primarily with examples from English, but expect that the mechanisms introduced here should be useful in building grammars for a wide variety of human languages.

10 Acknowledgments

This paper is an extension of work first reported in Flickinger and Bender 2003. We are grateful to the authors of *Minimal Recursion Semantics: An Introduction*, from which this documentation draws heavily, and especially to Ann Copestake for illumination on many issues presented here. We also thank the early adopters of the Matrix, in particular Melanie Siegel as principal author of the wide-coverage JACY grammar of Japanese, and the developers of the trail-blazing NorSource grammar of Norwegian: Lars Hellan, Dorothee Beermann, and Petter Haugereid, as well as their colleague Kaja Borthen. We have received helpful questions and critique from the other members of the Deep Thought consortium that helped to fund this work, especially from Berthold Crysmann, currently developing a wide-coverage grammar of German for the project; and also from colleagues working in the Norwegian LOGON machine translation project. As always, only the authors of this document can be held responsible for any errors that survive here in spite of the excellent counsel we received.

References

Bender, Emily, and Dan Flickinger. 1999. Peripheral constructions and core phenomena. In G. Webelhuth, A. Kathol, and J.-P. Koenig (Eds.), *Lexical and Constructional Aspects of Linguistic Explanation*. Stanford: CSLI Publications.

Bender, Emily M., Dan Flickinger, and Stephan Oepen. 2002. The Grammar Matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th Internation Conference on Computational Linguistics*, 8–14, Taipei, Taiwan.

Carpenter, Bob. 1992. *The Logic of Typed Feature Structures.* Cambridge, UK: Cambridge University Press.

Copestake, Ann. 2002. *Implementing Typed Feature Structure Grammars.* Stanford, CA: CSLI Publications.

Copestake, Ann. 2003. Report on the design of RMRS: Deep Thought project report D1.1. Unpublished ms.

Copestake, Ann, Daniel P. Flickinger, Ivan A. Sag, and Carl Pollard. 2003. Minimal Recursion Semantics. An introduction. Unpublished ms.

Copestake, Ann, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*, Toulouse, France.

Flickinger, Dan. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering* 6(1).

Flickinger, Dan, and Emily M. Bender. 2003. Compositional semantics in a multilingual grammar resource. In *Proceedings of the ESSLLI 2003 Workshop "Ideas and Strategies for Multilingual Grammar Development"*, Vienna, Austria.

Flickinger, Dan, and Emily M. Bender. forthcoming. The matrix: Complementation. Unpublished ms.

Ginzburg, Jonathan, and Ivan A. Sag. 2000. *Interrogative Investigations: The form, meaning and use of English interrogatives.* Stanford, CA: CSLI Publications.

Krieger, Hans-Ulrich, and Ulrich Schaefer. 1994. TDL - a type description language for HPSG. Research Report, Deutsches Forschungszentrum fuer Kuenstliche Intelligenz, Saarbruecken.

Oepen, Stephan, Daniel Flickinger, J. Tsujii, and Hans Uszkoreit (Eds.). 2002. *Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing.* Stanford, CA: CSLI Publications.

Pollard, Carl, and Ivan A. Sag. 1994. *Head-driven Phrase Structure Grammar.* Chicago: Chicago University Press.

Siegel, Melanie, and Emily M. Bender. 2002. Efficient deep processing of japanese. In *Proceedings of the 3rd Workshop on Asian Language Resources and Standardization at the 19th Internation Conference on Computational Linguistics*, Taipei, Taiwan.

Wahlster, Wolfgang, and Reinhard Karger. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation.* Berlin, Heidelberg, and New York: Springer Verlag.